Report No. 174

510.84
Il6r
no.174
cop.3

SOME THEOREMS AND ALGORITHMS ON A NEW FORM OF DECOMPOSITION FOR AUTOMATA

by

Arthur T. Pu

April 12, 1965

DEPARTMENT OF COMPUTER SCIENCE · UNIVERSITY OF ILLINOIS · URBANA, ILLINOIS

Report No. 174

SOME THEOREMS AND ALGORITHMS ON A NEW FORM OF DECOMPOSITION FOR AUTOMATA

by

Arthur T. Pu

April 12, 1965

Department of Computer Science
University of Illinois
Urbana, Illinois

ACKNOWLEDGMENT

The author wishes to thank Professor David E. Muller of the Department of Mathematics for his helpful suggestions.

INTRODUCTION

In recent years, due to the need to obtain economical realization of machines, especially large computers, intensive effort has been devoted in the theory of decomposition of machines. Based on criterion of reduced dependence, Hartmanis [4,5] and Hartmanis and Stearns [6,7] have published a series of articles on this subject. Probing through the properties of semi-groups, Krohn and Rhodes [9] were able to attain a rather general theory. However, their results and the results of most of the others [13,14] were based on the question of decomposition into series or parallel connections. The information is not allowed to interflow between blocks.

With the new technology, such as microelectronic technique, progressing so rapidly, the problem of economical realization no longer lies in the complexity of logical design in each building block. Rather it tends to depend on the number of connections between basic building blocks. The problems arising from future practical design thus will clearly emphasize the importance of having smaller number of connections.

With this in mind, the present work starts from an entirely new approach to the problem of decomposition. This approach allows information to interflow between blocks. Hopefully, some of the known results will be special cases of our theory. This report represents the result of preliminary investigation.

Part 1 is devoted to the basic theorems of decomposition. A decomposition of a machine $M = \langle S,T,f \rangle$ into two machines M_1, M_2; $M_1 = \langle A,T,\{f_b\} \rangle$, $M_2 = \langle B,T,\{f_a\} \rangle$ is defined as a one-to-one mapping φ from $S \to A \times B$ satisfying certain conditions. We prove that for any given n_1, n_2 there is at least one decomposition of an n-state machine into two machines with n_1, n_2 states each if and only if $n_1 n_2 \geq n$. We further define the notion of "strongly equivalence" between two decompositions of a given machine, and later give a necessary and sufficient condition. We introduce the concept of connecting lines between decomposed machines. Using the notion of covering, we show that the problem of determining the number of connecting lines becomes a covering problem. Finally, we introduce the new concept of "measure" and give a special example of such a measure.

In order to obtain a best decomposition with respect to a measure, we must have a method to generate all decompositions of a machine. In Part 2, we develop two algorithms. The first one generates without duplication all decompositions of a given machine up to strongly equivalence. The second algorithm generates without duplication all decompositions for a fixed pair of integers n_1, n_2. The above algorithms do not depend on the specific measure one is using, thus it can be used to obtain a best decomposition with respect to any measure.

1. Basic Theorems of Decomposition

Let us consider an automaton $M = \langle S,T,f\rangle$, where S denotes the set of states, T the set of input letters and f the transition function. Let the number of states in M be n. Let n_1, n_2 be two positive integers such that $n_1 n_2 \geq n$. Let A', B' be two sets of n_1, n_2 elements respectively. Then there exist one-to-one functions from $S \to A' \times B'$. Let φ be such a function. Let $A \subseteq A'$, $B \subseteq B'$ such that

$$A = \{a \in A' \mid (a,b) \in \text{Image } \varphi \text{ for some } b \in B'\}$$

$$B = \{b \in B' \mid (a,b) \in \text{Image } \varphi \text{ for some } a \in A'\}.$$

For each $a \in A$ and $b \in B$, define

$$B_a = \{b \in B \mid (a,b) \in \text{Image } \varphi\}$$

$$A_b = \{a \in A \mid (a,b) \in \text{Image } \varphi\}$$

for $(a,b) \in$ Image φ, let s_{ab} denote the state in S such that $\varphi(s_{ab}) = (a,b)$, and for each $t \in T$, let $(a^t,b^t) = \varphi(f(s_{ab},t))$. For each $a \in A$, define $f_a : B_a \times T \to B$ such that $f_a(b,t) = b^t$, for each $b \in B_a$ and $t \in T$. Similarly, we may define $f_b : A_b \times T \to A$ such that $f_b(a,t) = a^t$. It is easy to check that $\{f_a\}_{a\in A}$ and $\{f_b\}_{b\in B}$ are well defined. We now have the relation for each $s \in S$, $t \in T$

$$\varphi(f(s,t)) = (f_b(a,t), f_a(b,t))$$

where $\varphi(s) = (a,b)$.

Now suppose we consider the tuples: $\langle B,T,\{f_{a_i}\}\rangle$ and $\langle A,T,\{f_{b_j}\}\rangle$, we know that they form incomplete machines. Therefore, for each correspondence φ

from states S to $A \times B$, it induces a natural decomposition of machines into smaller machines (however, incomplete ones).

Figure 1 can be regarded as the block diagram of our decomposition. Each $a_i \in A$, $b_j \in B$ will be viewed as states, and $\{f_{a_i}\}$, $\{f_{b_j}\}$ are transition functions. Original inputs are fed into both parts. Although we can regard the parts after decomposition as machines by considering f_{a_i} or f_{b_j} as part of the input, we shall refer to them as partial machines in order to distinguish them from others.

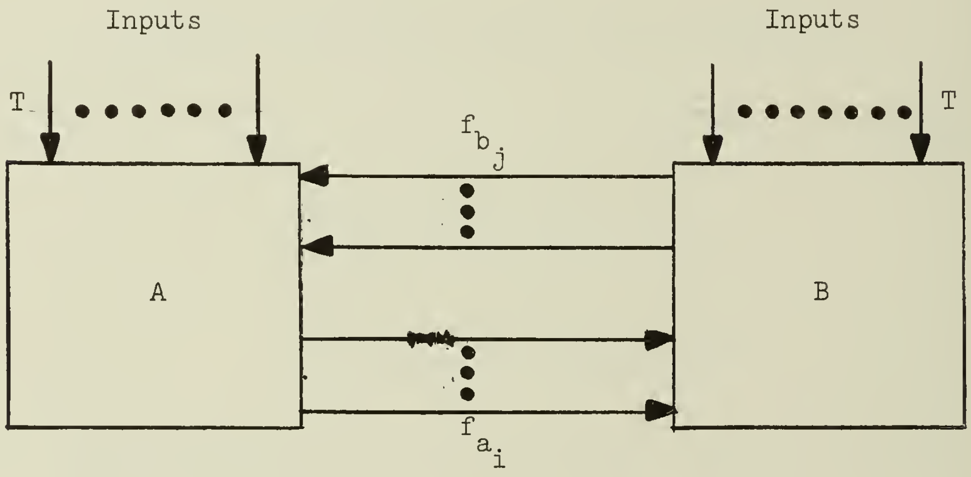

FIGURE 1

Definition 1. A decomposition of an n-state machine into two incomplete machines with n_1, n_2, states respectively where $n_1 n_2 \geq n$, is a one-to-one mapping φ from the n-tuple $(s_1, s_2, \ldots, s_n)$ to $(a_{i_1}, a_{i_2}, \ldots, a_{i_n}) \times (b_{j_1}, \ldots, b_{j_n})$ such that $\{1,2,3,\ldots,n_1\} = \{i_1, i_2, \ldots, i_n\}$ and $\{1,2,\ldots,n_2\} = \{j_1,\ldots,j_n\}$.

We remark that in the above definition we may consider $\varphi(s_k) = (a_{i_k}, b_{j_k})$ for each $s_k \in S$. Furthermore, for φ to be a decomposition we required that $A = A'$ and $B = B'$.

Theorem 1. For a given n-state machine M there is at least one decomposition into two incomplete machines with n_1, n_2 states respectively iff $n_1 n_2 \geq n$, $n_1 \leq n$, and $n_2 \leq n$.

Proof. Suppose $n_1 n_2 \geq n$ and $n_1 \leq n$, $n_2 \leq n$. Let $n_1 \geq n_2$. Let A, B be two sets with n_1, n_2 elements respectively;

$$A = \{a_1, a_2, \ldots, a_{n_1}\}$$

$$B = \{b_1, b_2, \ldots, b_{n_2}\}$$

The desired decomposition is

$$\varphi(s_1, s_2, \ldots, s_{n_2}, s_{n_2+1}, \ldots, s_{n_1}, \ldots, s_n)$$

$$= (a_{i_1}, a_{i_2}, \ldots, a_{i_n}) \times (b_{j_1}, b_{j_2}, \ldots, b_{j_n})$$

$$= (a_1, a_2, \ldots, a_{n_2}, a_{n_2+1}, \ldots, a_{n_1}, a_{p_1}, \ldots, a_{p_{n-n_1}}) \times (b_1, b_2, \ldots, b_{n_2}, b_{q_1}, \ldots, b_{q_{n-n_2}})$$

where b_{q_i}, a_{p_k} are chosen arbitrarily from B and A respectively such that $(a_{i_k}, b_{j_k}) \neq (a_{i_\ell}, b_{j_\ell})$ whenever $k \neq \ell$ and $1 \leq k, \ell \leq n$. Conversely, consider a decomposition φ of a given n-state machine M. Let n_1, n_2, be the numbers of distinct a_i, b_j respectively. Then $n_1 n_2 \geq n$; otherwise φ cannot be one-to-one. Since there are only n-states, none of n_1, n_2 can be bigger than n. QED

Definition 2. Let P_A be a permutation on $\{1,2,\ldots,n_1\}$ and P_B be a permutation on $\{1,2,\ldots,n_2\}$. We define $P_A \times P_B$ as an operation which sends $(a_{i_1}, \ldots, a_{i_n}) \times (b_{j_1}, \ldots, b_{j_n})$ to $(a_{P_A(i_1)}, \ldots, a_{P_A(i_n)}) \times (b_{P_B(j_1)}, \ldots, b_{P_B(j_n)})$.

Definition 3. Two decompositions φ_1, φ_2 of the same n-state machine are said to be strongly equivalent; $\varphi_1 \equiv \varphi_2$ if and only if n_1, n_2 are the same for both φ_1, φ_2 such that if $<A_1,B_1>$, $<A_2,B_2>$ are the decomposed machines, then A_1, A_2 are the same machine, and B_1, B_2 are the same machine (up to renaming the states and inputs).

We note that two n-state machines are the same if (1) there exists an assignment of states to distinct numbers such that their transition tables are the same, and (2) if for each input i in one machine there exists an input j in the other and vice versa such that their next state table are the same.

Lemma 1. Given two decompositions φ_1, φ_2 of an n-state machine with the same n_1 and n_2, if P_A is a transposition on a_{i_k}, a_{i_ℓ}, for some k, ℓ, $1 \leq k, \ell \leq n_1$, such that $\varphi_2 = (P_A \times I)\varphi_1$. Then $\varphi_1 \equiv \varphi_2$.

Proof. Let $\varphi_1(s_1,\ldots,s_n) = (a_{i_1},\ldots,a_{i_k},\ldots,a_{i_\ell},\ldots,a_{i_n}) \times (b_{j_1},\ldots,b_{j_n})$.
Then $(P_A \times I)\varphi_1(s_1,\ldots,s_n) = (a_{i_1},\ldots,a_{i_\ell},\ldots,a_{i_k},\ldots,a_{i_n}) \times (b_{j_1},\ldots,b_{j_n})$.

We shall prove the lemma in two parts.

(a) To show that A_1, A_2 are the same machine.

Consider A_1, A_2 as machines by regarding $B \times T$ as the set of input alphabets. In the transition table of A_1, we denote each state a_{i_m}, $1 \leq m \leq n_1$, by m. Thus, the states a_{i_k}, a_{i_ℓ} are denoted by k and ℓ respectively. In the transition table of A_2, we denote each state by the same number as in A_1, except a_{i_ℓ}, a_{i_k} will be denoted by k, ℓ respectively. Since $\varphi_2 = (P_A \times I)\varphi_1$ and P_A is a transposition on a_{i_k}, a_{i_ℓ}, the transition tables of machine A_1 and A_2 after the above denotation will be the same. Thus A_1, A_2 are the same machine.

(b) To show that B_1 and B_2 are the same machine.

In both the transition table of B_1 and B_2, let each state b_{j_m}, $1 \leq m \leq n_2$, be denoted by m. The transition tables of B_1, B_2 are the same except under the inputs $(I \times a_{i_k})$ and $(I \times a_{i_\ell})$ as I ranges over T. However, for each $(I \times a_{i_k})$ in B_1 there is an input $(I \times a_{i_\ell})$ in B_2 and vice versa such that their next states are the same. This is caused by the transposition P_A. Therefore, B_1, B_2 are the same machine. QED

Theorem 2. Two decompositions φ_1, φ_2, of the same n-state machine are strongly equivalent if and only if n_1, n_2, are the same for both decompositions such that there are two permutations P_A, P_B such that $\varphi_2 = (P_A \times P_B)\varphi_1$.

Proof. Suppose n_1, n_2 are the same for φ_1, φ_2 and there are P_A, P_B such that $\varphi_2 = (P_A \times P_B)\varphi_1$. Certainly $(P_A \times P_B)\varphi_1 = (P_A \times I)(I \times P_B)\varphi_1$. Since every permutation is a composition of transpositions, by repeated usage of Lemma 1, we have $\varphi_1 \equiv \varphi_2$.

Conversely, let $\varphi_1(s_1,\ldots,s_n) = (a_{i_1},\ldots,a_{i_n}) \times (b_{j_1},\ldots,b_{j_n})$ and $\varphi_2(s_1,\ldots,s_n) = (a_{k_1},\ldots,a_{k_n}) \times (b_{\ell_1},\ldots,b_{\ell_n})$ and that $\varphi_1 \equiv \varphi_2$. Now consider

$$\pi = \begin{pmatrix} a_{i_1},\ldots,a_{i_n} \\ a_{k_1},\ldots,a_{k_n} \end{pmatrix}$$

Since $\varphi_1 \equiv \varphi_2$ implies φ_1 and φ_2 have the same n_1 and n_2, thus there is exactly n_1 distinct a_i among them. Moreover since A_1, A_2 have the same transition table, it is not hard to see that if $a_{i_m} = a_{i_n}$, then $a_{k_m} = a_{k_n}$. Then let P_A be the n_1 distinct columns in

$$\begin{pmatrix} a_{i_1},\ldots,a_{i_n} \\ a_{k_1},\ldots,a_{k_n} \end{pmatrix}$$

Similarly, we obtain P_B. It is easily shown that $\varphi_2 = (P_A \times P_B)\varphi_1$. QED

Proposition 1. For a given n-state machine M, there is just one equivalence class of decomposition in which n_1 or n_2 is one.

Proof. Let φ_1, φ_2 be any two decompositions of M with $n_1 = 1$. By Theorem 1, n_2 is necessarily n. Let

$$\varphi_1(s_1,s_2,\ldots,s_n) = (a_1,\ldots,a_1) \times (b_{i_1},\ldots,b_{i_n})$$

$$\varphi_2(s_1,s_2,\ldots,s_n) = (a_1,\ldots,a_1) \times (b_{j_1},\ldots,b_{j_n})$$

We have $\{1,2,\ldots,n\} = \{i_1,i_2,\ldots,i_n\} = \{j_1,j_2,\ldots,j_n\}$.

Let $P_A = I$ = identity permutation and

$$P_B = \begin{pmatrix} i_1,i_2,\ldots,i_n \\ j_1,j_2,\ldots,j_n \end{pmatrix}$$

Then P_B is a permutation on $\{1,2,\ldots,n\}$. Clearly $\varphi_2 = (P_A \times P_B)\varphi_1$. Thus $\varphi_2 \equiv \varphi_1$ by Theorem 2. QED

Proposition 2. Given an n-state machine M, there is just one equivalence class of decomposition with $n_1 = n_2 = n$.

Proof. Let φ_1, φ_2, be two decompositions of M with $n_1 = n_2 = n$. Let

$$\varphi_1(s_1,\ldots,s_n) = (a_{i_1},\ldots,a_{i_n}) \times (b_{j_1},\ldots,b_{j_n})$$

and

$$\varphi_2(s_1,\ldots,s_n) = (a_{\ell_1},\ldots,a_{\ell_n}) \times (b_{k_1},\ldots,b_{k_n}).$$

We also have:

$$\{1,2,\ldots,n\} = \{i_1,\ldots,i_n\} = \{j_1,\ldots,j_n\} = \{\ell_1,\ldots,\ell_n\} = \{k_1,\ldots,k_n\}$$

Let

$$P_A = \begin{pmatrix} i_1, i_2, \ldots, i_n \\ \ell_1, \ell_2, \ldots, \ell_n \end{pmatrix}$$

$$P_B = \begin{pmatrix} j_1, j_2, \ldots, j_n \\ k_1, k_2, \ldots, k_n \end{pmatrix}$$

Then it is obvious that P_A, P_B are permutations on the set $\{1,2,\ldots,n\}$ and that $\varphi_2 = (P_A \times P_B)\varphi_1$. QED

Definition 4. For any two symbols a_1, a_2 of A, f_{a_1} is compatible with f_{a_2}; $f_{a_1} \sim f_{a_2}$ if and only if they are the same mapping over their common domain (i.e., $B_{a_1} \cap B_{a_2} \times T$).

We note that compatibility is not an equivalence relation. It is an equivalence relation only when $n_1 n_2 = n$, since in that case the common domain for all f_a is $B \times T$.

Suppose we consider a set of mutually compatible functions f_{a_i} as one function. As a_i ranges over A, we obtain a collection of such distinct functions. If t_i is the number of elements in this collection, denote by F_A the minimum t_i as we range over all such possible collections. Similarly, we define F_B. In view of the nature of f_{a_i} and f_{b_j}, it is reasonable to take $\lceil \log F_A \rceil$ and $\lceil \log F_B \rceil$ as the numbers of lines from A to B and from B to A respectively. For any nonnegative real number x, $\lceil x \rceil$ denotes the least nonnegative integer bigger or equal to x.

Here we shall introduce the covering problem. A cover Σ of a given set A is a collection of nonempty subsets A_k of A such that their union is A;

$$\Sigma = \{A_k | A_k \subset A, \bigsqcup_{A_k \in \Sigma} A_k = A\}$$

Any collection of elements in Σ having the same property, namely their union is A, is a subcover of the cover Σ. A minimum subcover Σ_M of a given cover Σ is a subcover such that it has the least cardinality. The classical covering problem is to find the minimum subcover of a given cover of a set.

Let $A = \{a_1, a_2, \ldots, a_{n_1}\}$. Let $\Sigma = \{A_k | A_k \subset A$, such that all $f_{a_i}, a_i \in A_k$, will be the same function over their common domain$\}$. If Σ_M is a minimum subcover of Σ, then the cardinality of Σ_M; card (Σ_M) is equal to F_A. Since a subcover Σ_1 of Σ is a subfamily of Σ such that for each f_{a_i}, $a_i \in A$, a_i is in some element of Σ_1, therefore if Σ_M is a minimum subcover, card $(\Sigma_M) = F_A$. Thus, the problem of finding F_A, F_B, which is the same as the problem of determining the number of connecting lines becomes a covering problem. This covering problem is the same problem as, for example, the state minimization of incomplete machines. Thus, a detailed discussion of this type of problem can be found in several articles [1,2,3,8,10,12].

As noted before, the compatibility is an equivalence relation only when $n_1 n_2 = n$. In this case, the numbers F_A, F_B are uniquely determined by the numbers of equivalence classes. This is the special case where the given cover Σ of A (or B) is itself the minimum cover.

As for the case when $n_1 n_2 > n$, we can still obtain a crude estimation on the number of connecting lines without going into the covering theorem as follows. For a given state $a_i \in A$ and input $I \in T$, we refer the set of all $a_j \in A$ which are next states of this state input pair (a_i, I) under all possible f_{b_k} as the successor set. Let L_A be the maximum number of distinct elements in all the successor sets as (a_i, I) ranges over $A \times T$.

Proposition 3. The number L_A is a lower bound on the number of input configurations from B to A (i.e., $\lceil \log L_A \rceil$ is a lower bound on the number of connecting lines from B to A).

Proof. The definition of L_A implies that there are at least L_A number of distinct f_{b_j}, since these f_{b_j} map the same input state pairs to distinct next state. Therefore L_A is a lower bound. QED

A similar result for L_B can also be obtained.

2. The Measure of a Machine

In the course of studying the theory of decomposition of machines, one natural question arises as to how one compares one decomposition against another. Here, we will introduce the notion of a "measure" of a machine, and we extend it to be "a measure of a decomposition."

Now we will show some required properties of a measure of a machine and its relationship to a measure of a decomposition. A measure should be a real-valued function which measures the complexity of a given machine. Certainly, it is a function of the number of states as well as the number of input alphabets. This measure should be able to extend to be a measure of a decomposition and there the number of connection lines reveals its importance.

For a single-state machine, the measure should equal zero. This is due to the fact that in reality it requires nothing to construct. If one puts two identical n-state machines side by side, the performance will be the same as one n-state machine. However, it is twice as complicated. Thus the measure for such a decomposition should be twice that of the original n-state machine. This also shows that the measure of a decomposition is the sum of the measures of each decomposed machine. Thus, for a decomposition which gives a single-state machine as a part, the decomposition will have a measure equal to that of the original measure of the machine. This also coincides with the actual fact, since it necessarily is a composition of a single-state machine and an original machine.

We remark that the number of connecting lines alone cannot be the measure of a decomposition. Once again, if one considers two identical n-state machines operating side by side, there is no connecting line, yet the decomposition is twice as complicated.

The following is the development of a special measure of a machine. This measure seems to be quite reasonable and it satisfies all the desired properties.

3. A Special Measure

For an n-state machine M of K input configurations and Q output configurations, the circuit realization in terms of a set of combinational logical devices and memory elements is given in Fig. 2.

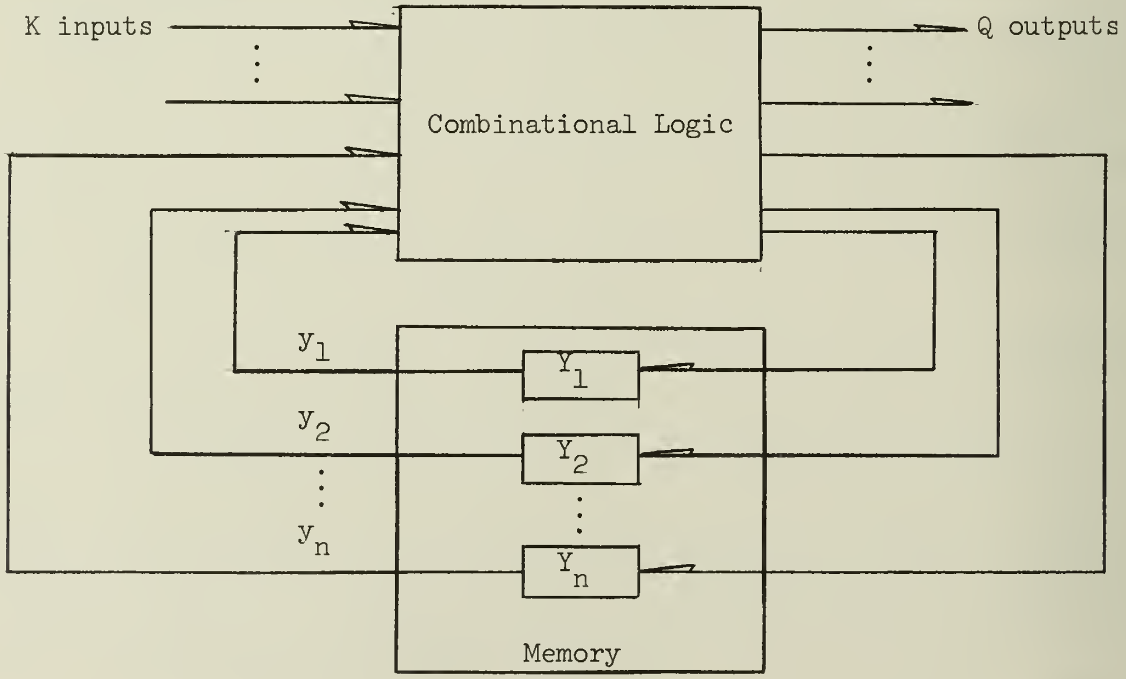

FIGURE 2

By a result of Muller [11], the maximal complexity for any electronic switching circuit of p inputs and q outputs is determined to be $\beta[2^r/r]$, where $r = p + \log q$ and β is a constant independent of p and q. Applying it to our case, for the combinational part, $p = \log k + \log n$ and $q = \log Q + \log n$. Then $r = \log nk + \log \log nQ$. Therefore, we have the complexity for the combinational part:

$$\beta[2^r/r] = \beta\left[\frac{nk \log Qn}{\log nk + \log \log nQ}\right]$$

Since β is a constant independent of p,q, we can eliminate the term $[\log \log nQ]$ in the above expression [11]. We shall define this complexity to be zero in the case when both denominator and numerator vanish. For the memory part, the complexity is defined as $\alpha[\log n]$, where α is a constant.

Definition 5. The measure $\mathcal{M}$ of an n-state machine with k inputs and Q outputs is:

$$\mathcal{M} = \alpha[\log n] + \beta\left[\frac{nk \log nQ}{\log nk}\right]$$

For nonprinting machines, Q is one; thus the measure for such machines is:

$$\alpha[\log n] + \beta\left[\frac{nk \log n}{\log nk}\right]$$

Definition 6. Two inputs t_1, t_2 of T are said to be A equivalent; $t_1 \overset{A}{\equiv} t_2$, if and only if for all $a \in A$, $f_a(b,t_1) = f_a(b,t_2)$ for all b whenever the equation makes sense.

Definition 7. If a_1, a_2, in A, then a_1 is said to be equivalent to a_2; $a_1 \overset{A}{\equiv} a_2$ iff $f_{a_1} \sim f_{a_2}$ and for all $b \in B$, $f_b(a_1,t) = f_b(a_2,t)$ for all $t \in T$, whenever the equation makes sense.

Similarly, we can define $t_1 \overset{B}{\equiv} t_2$ and $b_1 = b_2$.

Suppose φ is a decomposition of a given n-state machine M. (We shall always consider nonprinting machines unless otherwise stated.) Let n_A be the number of distinct a_i under "$\overset{A}{\equiv}$" and T_A the number of distinct t under "$\equiv$". We denote similarly for n_B, T_B. Using Definition 5 the measure $\mathcal{M}_A$ for part A in Fig. 1

$$\mathcal{M}_A = \alpha[\log n_A] + \beta\left[\frac{n_A T_A F_B \log n_A}{\log n_A T_A F_B}\right]$$

Correspondingly, the measure $\mathcal{M}_B$ for part B is:

$$\mathcal{M}_B = \alpha[\log n_B] + \beta\left[\frac{n_B T_B F_A \log n_B}{\log n_B T_B F_A}\right]$$

Definition 8. The measure of a decomposition φ is defined as the sum of the measure of part A and part B.

Now we proceed to show that this measure does have the desired property.

Proposition 4. Let $\mathcal{M}_M$ be the measure of a given n-state machine M. If φ is any decomposition of M into two machines $\langle M_1, M_2 \rangle$ such that $M_1 = M = M_2$, then $\mathcal{M}_\varphi = 2\mathcal{M}_M$, where $\mathcal{M}_\varphi$ denotes the measure of φ.

Proof. By Proposition 2, there is essentially one type of decomposition with $n_1 = n_2 = n$. Let the machine M be of K inputs. Then by Definition 5,

$$\mathcal{M}_M = \alpha[\log n] + \beta\left[\frac{nK \log n}{\log nK}\right]$$

For the decomposition φ, it can be shown that $F_A = F_B = 1$, $T_A = T_B = K$ and $n_A = n_B = n$. Readily, we have:

$$\mathcal{M}_\varphi = 2\left\{\alpha[\log n] + \beta\left[\frac{nK \log n}{\log nK}\right]\right\} = 2\mathcal{M}_M$$

QED

<u>Proposition 5</u>. If φ is any decomposition of a given n-state machine M into two machines $<M_1,M_2>$ such that M_1 is a 1-state machine, then $\mathcal{M}_\varphi = \mathcal{M}_M$.

<u>Proof</u>. If M_1 is of 1-state machine, by Theorem 1, M_2 must be of n-state machine. By Proposition 1, there is essentially one type of decomposition with $n_1 = 1$, $n_2 = n$. Again, assume M is of K inputs. Then it can be easily shown that for decomposition φ, $n_A = 1$, $n_B = n$. $F_A = F_B = 1$, $T_A = 1$ and $T_B = K$. Hence,

$$\mathcal{M}_\varphi = \alpha[\log n] + \beta\left[\frac{nK \log n}{\log nK}\right] = \mathcal{M}_M$$

QED

In addition to the desired properties, our special choice of measure has another feature. Let an n-state machine be such that $n = n_1 n_2$. Consider the worst case in which $n_A = n_1$, $n_B = n_2$, $F_B = n_2$, $F_A = n_1$, $T_A = K = T_B$. This is the case where nothing can be reduced under the defined equivalence relations. Then the measure of this decomposition is:

$$M = \alpha[\log n_1] + \beta\left[\frac{n_1Kn_2 \log n_1}{\log n_1Kn_2}\right] + \alpha[\log n_2] + \beta\left[\frac{n_2Kn_1 \log n_2}{\log n_2Kn_1}\right]$$

$$= \alpha[\log n_1 + \log n_2] + \beta\left[\frac{n_1n_2K(\log n_1 + \log n_2)}{\log nK}\right]$$

$$= \alpha[\log n] + \beta\left[\frac{nK \log n}{\log nK}\right]$$

This particular decomposition has a measure equal to that of the original machine. It shows that the measure of any decomposition with $n_1n_2 = n$ cannot be greater than that of the original machine, since this is the worst situation that we can have. In fact, it can be stated more rigorously.

Proposition 6. For any nonprinting n-state machine M with K input, any decomposition φ of M with $n = n_1 \cdot n_2$ have measure equal to or less than that of the original machine.

Proof. Let $n = n_1n_2$ with $n_1 \geq n_2$. We will consider the worst case. We know that the maximal number of distinct compatible functions is limited by the number of elements in the domain. In fact, for B part the number F_B is limited by

$$F_B = \mathrm{Min}[n_2^{n_2 I}, n_1].$$

Similarly for F_A. Since $n_1 \geq n_2$, $F_A = n_2$, then the measure for the decomposition in the worst case is

$$\mathcal{M}_\varphi = [\log n_1] + \alpha[\log n_2] + \beta\left[\frac{n_1Kn_2 \log n_1}{\log n_1Kn_2}\right] + \beta\left[\frac{n_2KF_B \log n_2}{\log n_2KF_B}\right]$$

$$= \alpha[\log n] + \beta\left[\frac{nK \log n_1}{\log nK}\right] + \beta\left[\frac{n_2KF_B \log n_2}{\log n_2KF_B}\right]$$

Thus, by replacing F_B by n_1 and by the fact that $\frac{x}{\log x}$ is an increasing function for $x \geq 2$, we have

$$\left[\frac{n_2 K F_B \log n_2}{\log n_2 K F_B}\right] \leq \left[\frac{nk \log n_2}{\log nK}\right]$$

Therefore we have

$$\mathcal{m}_\varphi \leq \alpha[\log n] + \beta\left[\frac{nK \log n}{\log nK}\right] = \mathcal{m}_M$$

QED

A best decomposition of a given machine is defined to be the one which gives the least measure. Now, suppose we are interested in finding a best decomposition of a machine with respect to our special measure. Since $1 \leq n_1 \leq n$, $1 \leq n_2 \leq n$ and $n_1 n_2 \geq n$, there are only finitely many choices of n_1 and n_2. Thus, it is possible to find the best decomposition. For our choice of measure, the numbers n_A, n_B, F_A, F_B, T_A, T_B play the most important part. Yet, once the decomposition is given, these numbers are uniquely defined. Thus, the problem of finding a best decomposition becomes a problem of finding all possible decompositions for a machine. This later task, if successful, will be independent of the choice of measure. Therefore, for any reasonable choice of measure, this algorithm will be able to give the best decomposition according to that measure.

1. An Algorithm for Generating All Decompositions without Duplication

Consider an array of $n \times n$ squares with rows labeled a_i, $1 \leq i \leq n$, and columns labeled b_j, $1 \leq j \leq n$. They are labeled in consecutive order. Then a square in a_ith row and b_jth column is labeled (a_i, b_j). Suppose we denote each state in an n-state machine by a number in an arbitrary way from 1 to n. Each placement of these n numbers into the $n \times n$ array of squares is a one-to-one mapping φ from $\{1,2,\ldots,n\}$ to $A \times B$ in such a way that $\varphi(i)$ is equal to the label of the square in which the number i is placed. Then φ is indeed a decomposition of the n-state machine.

Definition 9. The kth rectangle is a $k_1 \times k_2$ array of squares in which k squares are occupied such that at least one of the k occupied squares lies in each row and each column.

Definition 10. If the kth rectange is a $k_1 \times k_2$ array of squares, then the kth neighboring rectangle is a $(k_1 + 1) \times (k_2 + 1)$ array of squares.

In order to make possible a procedure to generate decompositions without duplication, we shall order the squares in this $n \times n$ array of squares. There are many ways of doing this. However, we shall make the following choice for ordering these squares. The ordering will be done during the algorithm.

> The square at the upper left-hand corner will have the least order. As all squares of kth rectange are orders, order the kth neighboring rectange as follows: first order the new column (if any) from up to down, then order the new row from left to right. The square that is common to both the new column and the new row will have the highest order.

Definition 11. A state j is said to be moved in order in the kth (neighboring) rectangle if the state j is moved step by step into unoccupied squares of the kth (neighboring) rectangle according to the order of the squares. State j is not allowable to be moved into any square that state j has occupied previously.

The algorithm starts with certain initial decomposition. The initial decomposition is obtained as follows:

> State 1 is put into the square at the upper left-hand corner. After putting state i, the unordered squares of the ith neighboring rectangle will be ordered according to the method mentioned before. Then state (i + 1) is put into any unoccupied square of the ith neighboring rectangle.

As this process terminates we obtain an nth rectangle. All squares in the nth neighboring rectangle have been ordered. If the nth rectangle is $n_1 \times n_2$, we have a decomposition of this n-state machine into two partial machines with n_1, n_2 states each.

In each move in the following steps, if there are unordered squares appearing in the neighboring rectangle, we shall order them according to the ordering method mentioned before. However, the ordering will not exceed the boundary of this $n \times n$ array of squares.

Step 1. State n is moved in order in the $(n - 1)$th neighboring rectangle. Each move results in a new decomposition. After state n has occupied the square having the highest order in the $(n - 1)$th neighboring rectange, execute the next step.

Step i. (A_1) State $(n - i + 1)$ is moved in order in the $(n - i)$th neighboring rectangle.

(A_2) For each move in (A_1), state $(n - i + 2)$ is moved in order in the new $(n - i + 1)$th neighboring rectangle.

⋮

(A_j) For each move in (A_{j-1}), state $(n - i + j)$ is moved in order in the new $(n - i + j - 1)$th neighboring rectangle.

⋮

(A_i) For each move in (A_{i-1}), state n is moved in order in the new $(n - 1)$th neighboring rectangle.

Each sequence of execution from (A_1) to (A_i) results in a new decomposition. After each state $(n - i + j)$, $1 \leq j \leq i$, reaches the perspective highest orders in the perspective neighboring rectangles, execute the next step.

We will now proceed to show that the above algorithm will generate all decompositions of an n-state machine without duplication.

<u>Proposition 7</u>. From each strongly equivalence class of decomposition, there is a representative of the following form:

1. The state 1 is in the upper left-hand corner.
2. State 2 is in one of the unoccupied squares in the first neighboring rectangle.
3. After state i is obtained, state (i + 1) is in one of the unoccupied squares in the ith neighboring rectangle.

<u>Proof</u>. As a result of Theorem 2, any permutation among a_i (or b_j) will give strongly equivalent decompositions. Thus each equivalence class will have a representation in the above form. QED

From here on, we refer to the above form of a decomposition to be the <u>normal</u> form.

For a given decomposition φ in the normal form, for each $a_i \in A$, let $\min \varphi^{-1}(a_i)$ denote the minimum number k such that $\varphi(s_k) = (a_i, b_j)$ for any $b_j \in B$.

<u>Proposition 8</u>. For any given a_i, $a_j \in A$, $i < j$ if and only if $\min \varphi^{-1}(a_i) < \min \varphi^{-1}(a_j)$.

<u>Proof</u>. (1) Let $a_i, a_j \in A$ such that $i < j$. Since φ is in the normal form, at least one state is in a_ith row before any later state is in a_jth row. Thus $\min \varphi^{-1}(a_i) < \min \varphi^{-1}(a_j)$.

(2) Conversely, let $\min \varphi^{-1}(a_i) < \min \varphi^{-1}(a_j)$ and $i \not< j$. Then $i \geq j$. Yet, by the proof of (1), $i \geq j$ implies $\min \varphi^{-1}(a_i) \geq \min \varphi^{-1}(a_j)$ which leads to a contradiction. Hence if $\min \varphi^{-1}(a_i) < \min \varphi^{-1}(a_j)$, then $i < j$. QED

Proposition 9. From each strongly equivalence class, there exists a unique representative in the normal form.

Proof. Suppose there are two decompositions φ_1, φ_2 both of normal form and $\varphi_1 \equiv \varphi_2$. Then there exists permutations P_A, P_B such that $(P_A \times P_B)\varphi_1 = \varphi_2$. If $P_A \neq I$, then it is at least a transposition on some letters a_i, a_j. Without loss of generality, assume $i < j$. By Proposition 8, $\min \varphi_1^{-1}(a_i) < \min \varphi_1^{-1}(a_j)$. However, $\min \varphi_2^{-1}(a_i) = \min \varphi_1^{-1}(a_j)$ and $\min \varphi_2^{-1}(a_j) = \min \varphi_1^{-1}(a_i)$. Therefore, we have $i < j$ implies $\min \varphi_2^{-1}(a_i) > \min \varphi_2^{-1}(a_j)$, which contradicts the assumption that φ_2 is in the normal form. Therefore, $P_A = I$. Similarly, we can show $P_B = I$. Hence the representative in normal form from each strongly equivalence class is unique. QED

Theorem 3. All decompositions of a given n-statemachine M are generated without duplication from the above algorithm.

Proof. First, we note that the algorithm will terminate. The ordering of the squares is done during the process of the algorithm. The size of the nth rectangle is expanding until it reaches the boundary of $n \times n$ array of squares. Eventually, only squares inside the $n \times n$ array of squares are ordered. Thus, the algorithm terminates as soon as the states exhaust all finite possible moves.

In the construction of the algorithm, we require state 1 to be at the upper left-hand corner. For each state $(i + 1)$, it is in one of the squares in the ith neighboring rectangle. Therefore, only decompositions of the normal form are generated.

Now, we shall prove by induction on the number of states that each decomposition φ in the normal form is generated. Suppose φ is such a decomposition in the normal form for a two-state machine. Then φ is of one of the following three types: (1) $n_1 = 1$, $n_2 = 2$; (2) $n_1 = 2$, $n_2 = 2$; (3) $n_1 = 2$, $n_2 = 1$. In the algorithm, state 1 is fixed, and there are three unoccupied but ordered squares in the first neighboring rectangle. However, state 2 is moved in order in these three unoccupied squares. Hence, no matter which one of the three type φ is, φ is obtained. Suppose this is true for any j-state machine. Namely, so long as φ is of the normal form, the positions of each of the j states in the $j \times j$ array of squares will be reached by the algorithm. Consider the first j states of a given $(j + 1)$-state machine. These j states are inside the $j \times j$ array of squares. By induction hypothesis, the positions of these states can be reached. State $(j + 1)$ is in any unoccupied but ordered square in the jth neighboring rectangle. Since state $(j + 1)$ is moved in order, the position of the $(j + 1)$ state under φ can also be reached by the algorithm. Hence, for any n-state machine, this algorithm generates all decompositions of the normal form.

Since by Proposition 7, each strongly equivalence class contains a representative in the normal form, therefore, this algorithm essentially generates all decompositions of a given n-state machine.

From Proposition 9, the representation in normal form from each strongly equivalence class is unique, hence this algorithm generates all decompositions without duplication. QED

Remark 1. The $n \times n$ array of squares is symmetric with respect to the main diagonal. Therefore, although the algorithm generates no duplication, it will generate two decompositions $\varphi_1 = <A_1,B_1>$, $\varphi_2 = <A_2,B_2>$ such that A_1,B_2 are the same and B_1,A_2 are the same. This can be eliminated if we put more restriction in the algorithm.

We can make modifications to the above algorithm such that only decompositions whose n_1, n_2 are less than or equal to some fixed numbers n_1', n_2' respectively, will be generated without duplication.

The method of ordering the squares will be the same except that only the squares inside $n_1' \times n_2'$ array of squares will be eventually ordered. We require the initial decomposition to have an nth rectangle less than or equal to $n_1' \times n_2'$. The algorithm is exactly the same as before. Since the ordering will not continue in that direction once it reaches the boundary n_1' (or n_2'), the proof for generation without duplication of all decompositions whose n_1, n_2 are less than or equal to n_1', n_2' respectively is the same as before.

2. An Algorithm for Generating without Duplication All Decompositions with n_1, n_2 Fixed

The following is the construction of an algorithm which generates only those decompositions whose n_1, n_2 are fixed ($n_1 n_2 \geq n$). These are decompositions whose nth rectangles are exactly $n_1 \times n_2$.

Suppose we are putting n states into an $n_1 \times n_2$ array of squares. State 1 is put into the square at the upper left-hand corner. Suppose state i is put into some unoccupied square in the (i - 1)th neighboring rectangle, and the resulting ith rectangle is $n_{1,i} \times n_{2,i}$.

Proposition 10. If $n - i \geq n_2 - n_{2,i} \geq 0$ and $n - i \geq n_1 - n_{1,i} \geq 0$, then state $(i + 1)$ can be put into some squares in the ith neighboring rectangle such that $n - (i + 1) \geq n_2 - n_{2,i+1} \geq 0$ and $n - (i + 1) \geq n_1 - n_{1,i+1} \geq 0$.

Proof. First, we shall describe the method. If $n - i > n_2 - n_{2,i} \geq 0$ and $n - i > n_1 - n_{1,i} \geq 0$, then state $(i + 1)$ is put into any unoccupied square in the ith neighboring rectangle. If $n - i = n_1 - n_{1,i} > 0$ (or $n - i = n_2 - n_{2,i} > 0$), state $(i + 1)$ must be put into the $(n_{1,i} + 1)$th row (or $(n_{2,i} + 1)$th column).

Let the resulting $(i + 1)$th rectangle be $(n_{1,i+1} \times n_{2,i+1})$. Certainly $n_{1,i+1} \geq n_{1,i}$ and $n_{2,i+1} \geq n_{2,i}$

(a) if $n - i > n_2 - n_{2,i} > 0$ and $n - i > n_1 - n_{1,i} > 0$. We have $n - i - 1 > n_2 - n_{2,i} - 1 \geq 0$. Since these integers, $n - i - 1 \geq n_2 - n_{2,i} > 0$. Therefore $n - (i + 1) \geq n_2 - n_{2,i+1} \geq 0$. Similarly, $n - (i + 1) \geq n_1 - n_{1,i+1} \geq 0$.

(b) if $n - i > n_2 - n_{2,i} = 0$ and $n - i > n_1 - n_{1,i} \geq 0$. We have $n - i - 1 \geq n_2 - n_{2,i} = 0$. Since $n_2 = n_{2,i}$, then $n_{2,i} = n_{2,i+1} = n_2$. Therefore $n - (i + 1) \geq n_2 - n_{2,i+1} = 0$. If $n - i > n_1 - n_{1,i} = 0$, we can similarly obtain $n - (i + 1) \geq n_1 - n_{1,i+1} = 0$. If $n - i > n_1 - n_{1,i} > 0$, we can obtain $n - (i + 1) \geq n_1 - n_{1,i+1} \geq 0$ by (a).

(c) if $n - i = n_1 - n_{1,i} \geq 0$ (or $n - i = n_2 - n_{2,i} \geq 0$). In this case, $n_{1,i+1} = n_{1,i} + 1$. Thus $n - i - 1 = n_1 - n_{1,i} - 1 \geq 0$.
Hence $n - (i + 1) = n_1 - (n_{1,i} + 1) = n_1 - n_{1,i+1} \geq 0$. QED

<u>Proposition 11</u>. After the state n is in some square (using the method in Proposition 10) we obtain a decomposition of this n-state machine into two partial machines with states n_1 and n_2 each.

<u>Proof</u>. By Proposition 10 we know for each state i, $n - i \geq n_1 - n_{1,i} \geq 0$ and $n - i \geq n_2 - n_{2,i} \geq 0$. Thus, when $i = n$, we have $0 \geq n_1 - n_{1,i} \geq 0$ and $0 \geq n_2 - n_{2,i} \geq 0$. Hence $n_{1,n} = n_1$ and $n_{2,n} = n_2$. Therefore, the nth rectangle is exactly $n_1 \times n_2$. QED

<u>Definition 12</u>. A state $(i + 1)$ is said to be moved <u>in order</u> into <u>an allowed square</u> in the ith neighboring rectangle if state $(i + 1)$ is moved <u>in order</u> such that

(1) when $n - i = n_2 - n_{2,i}$ (or $n - i = n_1 - n_{1,i}$), this square lies in the $(n_{2,i} + 1)$th column (or in the $(n_{1,i} + 1)$th row).

(2) when $n - i > n_1 - n_{1,i}$ and $n - i > n_2 - n_{2,i}$, this square lies in the ith neighboring rectangle.

We shall start the algorithm with an initial decomposition whose nth rectangle is exactly $n_1 \times n_2$. This is done by the method described in Proposition 10. The ordering of the squares is as before. Once the initial decomposition is chosen, all n_1n_2 squares are ordered.

Let $(n - i)$ be the smallest number such that for state $(n - i)$, $n - (n - i) = n_1 - n_{1,n-i}$ and $n - (n - i) = n_2 - n_{2,n-i}$.

<u>Step 1</u>. State $(n - i)$ is moved in order into an allowed square in the $(n - i - 1)$th neighboring rectangle. Each move results in a new decomposition. After state $(n - i)$ reaches the allowed square of the highest order, execute the next step.

<u>Step j</u>. (A_1) State $(n - i - j)$ is moved in order into an allowed square in the $(n - i - j - 1)$th neighboring rectangle.

⋮

(A_k) For each move in (A_{k-1}), state $(n - i - j + k - 1)$ is moved in order into an allowed square in the new $(n - i - j + k - 2)$th neighboring rectangle.

⋮

(A_{i+j+1}) For each move in (A_{i+j}), state n is moved in in order into an allowed square in the new $(n - 1)$th neighboring rectangle.

Each sequence of move from (A_1) to (A_{i+j+1}) results in a new decomposition. After each state reaches its perspective allowed squares of highest order, execute the next step.

Theorem 4. For any given n-state machine M, if n_1, n_2 are given fixed numbers such that $n_1 n_2 \geq n$, then all decompositions into machines with exactly n_1, n_2 state each are generated without duplication by the above algorithm.

Proof. The algorithm will terminate since there are only finitely many allowed squares for each state.

By Proposition 10 and Proposition 11, we know that each decomposition obtained from the algorithm will have the nth rectangle exactly equal to $n_1 \times n_2$. Since these decompositions are also obtainable from Theorem 3, only the representatives in the normal form of the strongly equivalence classes are generated.

Suppose for a given decomposition φ, the procedure outlined in Proposition 10 is violated for some state i. Then if $n_{1,i} \times n_{2,i}$ is the resulting ith rectange, we have $n - i < n_1 - n_{1,i} \geq 0$ or $n - i < n_2 - n_{2,i} \geq 0$. Suppose $n - i < n_1 - n_{1,i} \geq 0$. No matter how one puts the rest states $(i + 1), (i + 2), \ldots, n$, we have $n_{1,k} \leq n_{k,i-1} + 1$, where $i + 1 \leq k \leq n$. Hence $n_{1,i+j} \leq n_{1,i} + j$ for $1 \leq j \leq n - i$. Therefore $n_1 - n_{1,i+j} \geq n_1 - (n_{1,i} + j)$. Yet $n - (i + j) < n_1 - (n_{1,i} + j)$. Thus $n - (i + j) < n_1 - n_{1,i+j}$ for $1 \leq j \leq n - i$. For $j = n - i$, we have $0 < n_1 - n_{1,n}$. Therefore $n_{1,n} \neq n_1$. Hence any violation in the procedure will not generate a decomposition of the desired typed. Since the representative in the normal form from each strongly equivalence class is unique (Proposition 9), all decompositions into partial machines with n_1, n_2 state each are generated without duplication. QED

Remark 2. When $n_1 = n_2$, the case discussed in Remark 1 will occur. If $n_1 \neq n_2$, then $n_1 \times n_2$ array of squares is not symmetric with respect to the main diagonal. Therefore all decompositions are distinct even if we do distinguish the machine on the right from the left.

REFERENCES

[1] Beatty, J. and R. E. Miller. "Some Theorems for Incompletely Specified Sequential Machines with Applications to State Minimization." AIEE Proceedings of the Third Annual Symposium on Switching Circuit Theory and Logical Design, pp. 123-136, September 1962.

[2] Ginsburg, S. "A Technique for the Reduction of a Given Machine to a Minimal State Machine." Trans. IRE, EC-8, vol. 3, pp. 346-355, September 1959.

[3] Ginsburg, S. "Synthesis of Minimal State Machine." Trans. IRE, EC-8, vol. 4, pp. 441-449, December 1959.

[4] Hartmanis, J. "Symbolic Analyses of a Decomposition of Information Processing Machines." Information and Control, 3, pp. 154-178, June 1960.

[5] Hartmanis, J. "On the State Assignment Problem for Sequential Machine I." Trans. IRE, EC-10, pp. 157-165, June 1961.

[6] Hartmanis, J. and R. E. Stearns. "On the State Assignment Problem of Sequential Machine II." Trans. IRE, EC-10, pp. 593-603, December 1961.

[7] Hartmanis, J. and R. E. Stearns. "Pair Algebra and Its Application to Automata Theory." Information and Control, 7, pp. 485-507, December 1964.

[8] Karp, R. "Some Techniques of State Assignment for Synchronous Sequential Machines." IBM Report RC-938, May 1963.

[9] Krohn, K. and J. L. Rhodes. "Algebraic Theory of Machines I, II, III." Proceedings of the Symposium on Mathematical Theory of Automata, Brooklyn, Polytechnique Institute, April 1962.

[10] Miller, R. E. "State Reduction for Sequential Machines." IBM Research Report RC-121, June 15, 1959.

[11] Muller, D. E. "Complexity in Electronic Switching Circuits." Trans. IRE, EC-5, March 1956.

[12] Paull, M. C. and S. H. Unger. "Minimizing the Number of States in Incompletely Specified Sequential Switching Functions." Trans. IRE, EC-10, pp. 356-367, September 1959.

[13] Yoeli, M. "The Cascade Decomposition of Sequential Machines." Trans. IRE, EC-10, pp. 587-592, December 1961.

[14] Yoeli, M. "Cascade-Parallel Decomposition of Sequential Machines." Trans. IRE, EC-12, pp. 322-323, June 1963.